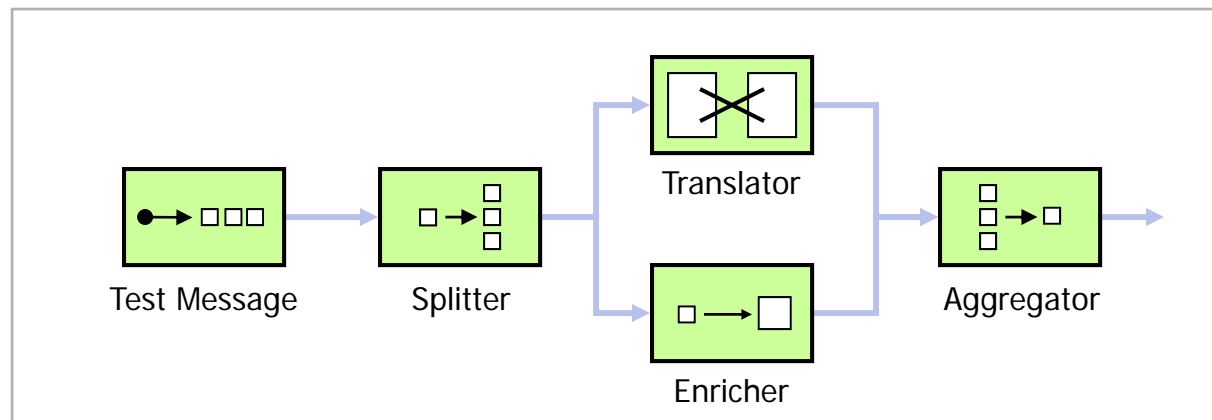




Enterprise Integration Patterns: Past, Present and Future

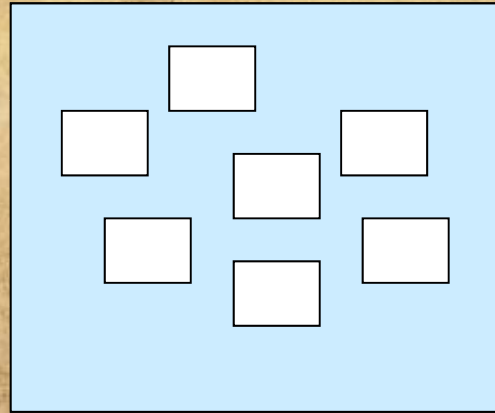


Gregor Hohpe
www.eaipatterns.com

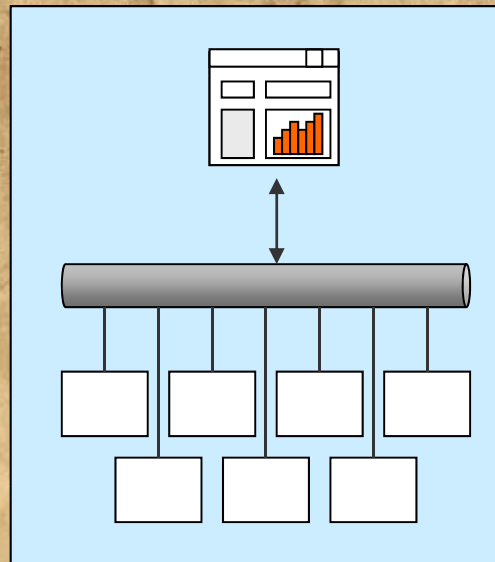


Live from
the Archives

Isolated Systems



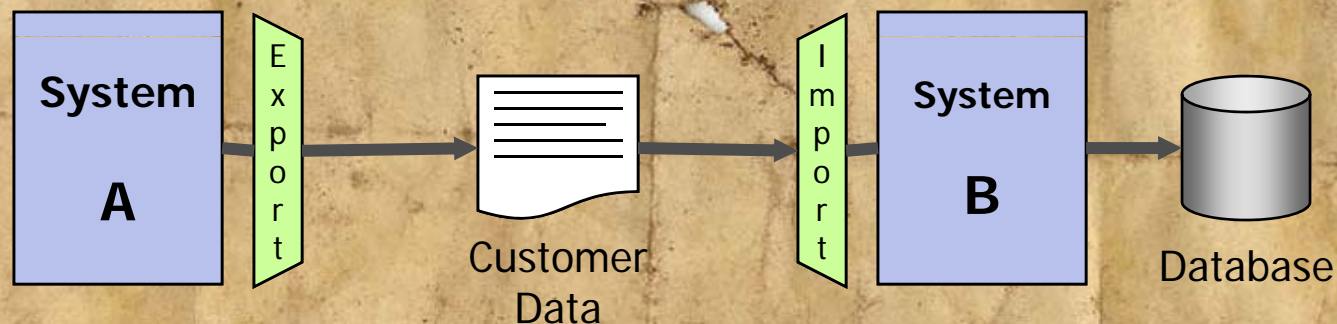
Unified Access



70s: Batch Data Exchange

Live from
the Archives

- Export information into a common file format, read into the target system
- Example: COBOL Flat files



Pros:

- Good physical decoupling
- Language and system independent

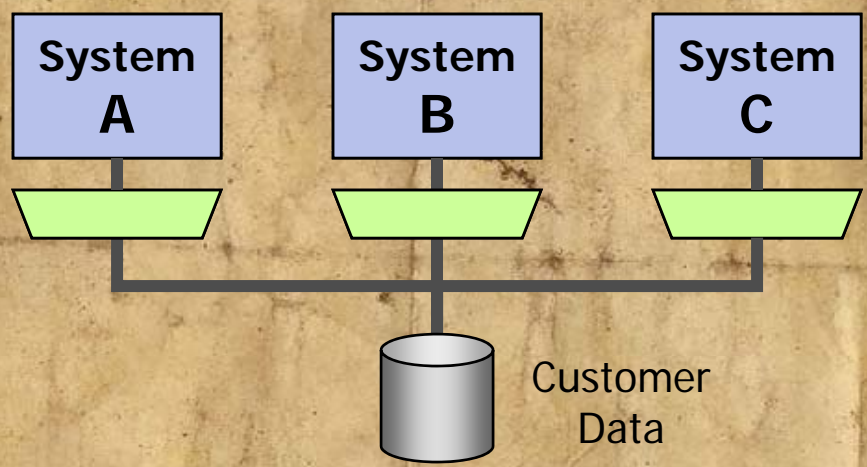
Cons:

- Data transfer not immediate
- Systems may be out of sync
- Large amounts of data

80s: Central Database

Live from the Archives

- All applications access a common database



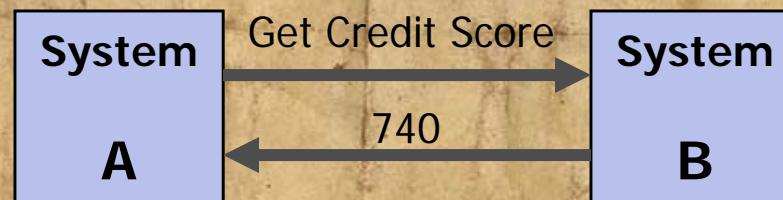
- Pros:
- Consistent Data
 - Reporting
 - Transactional guarantees

- Cons:
- Integration of data, not business functions
 - Difficult to find common representation

90s: Remote Procedure Calls

Live from
the Archives

- One application calls another directly to perform a function.
- Data necessary for the call is passed along. Results are returned to calling application.



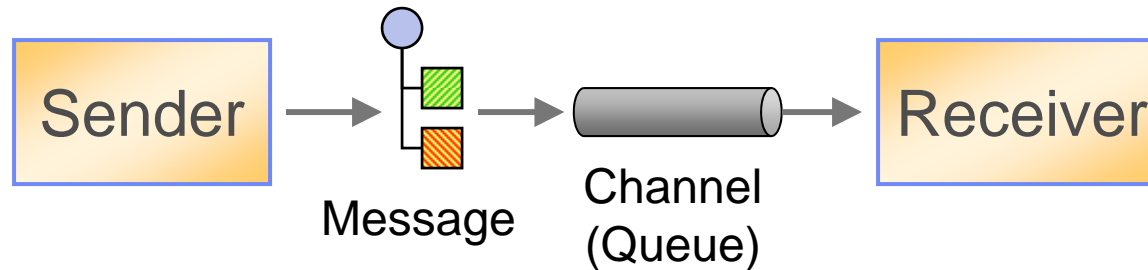
Pros:

- Data exchanged only as needed
- Integration of business function, not just data

Cons:

- Works well only with small number of systems
- Fragile (tight coupling)
- Performance

Asynchronous Messaging Style



- Systems send messages across Channels
- Channels have logical (location-indep.) addresses
- Placing a message into the Channel is quick ("fire-and-forget")
- The Channel queues messages until the receiving application is ready

Simplified
Interaction

Location
Decoupling

Temporal
Decoupling

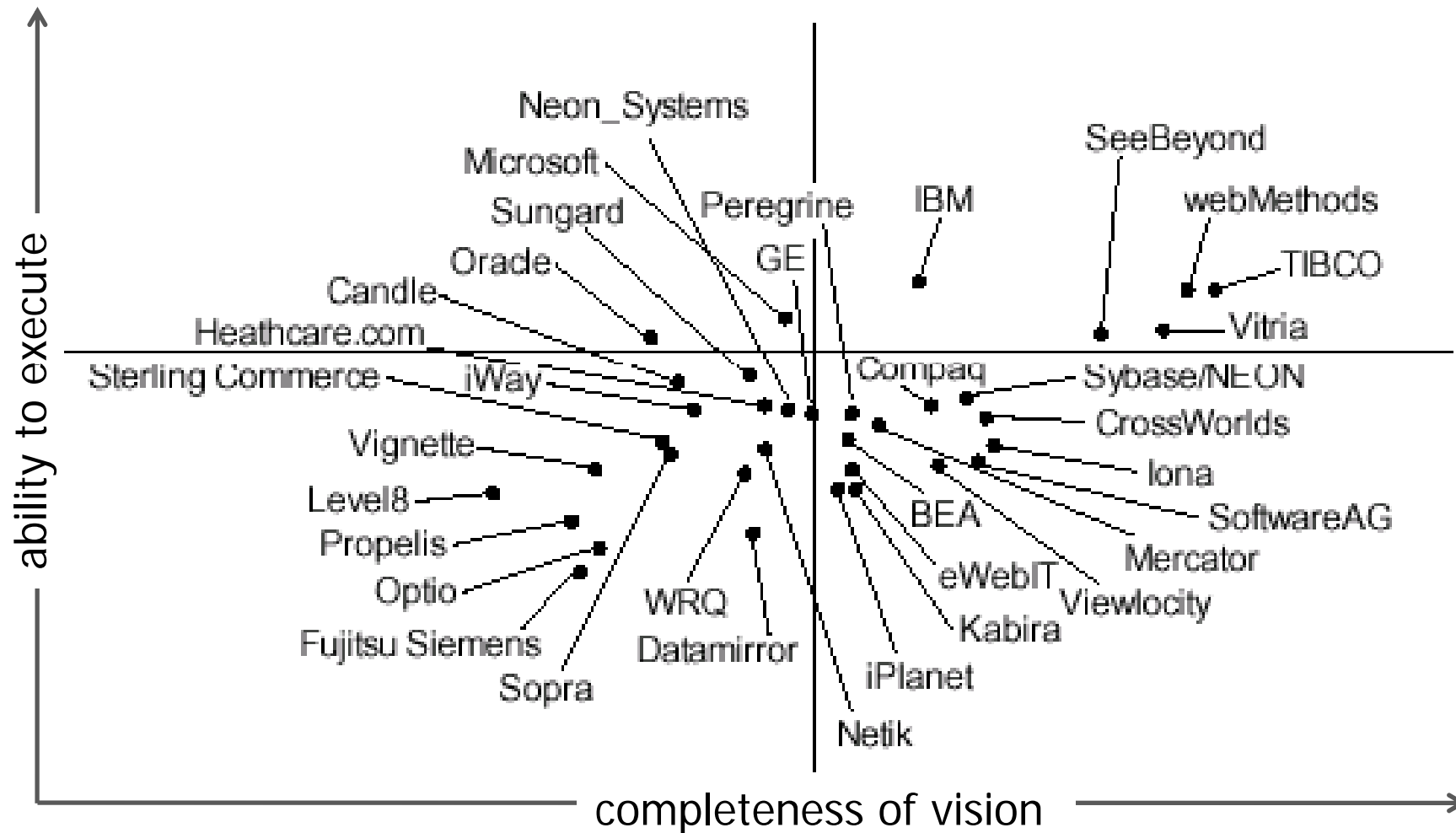
An "honest" architectural style that does not try to deny the limitations of the underlying medium.

Why Asynchronous Messaging?

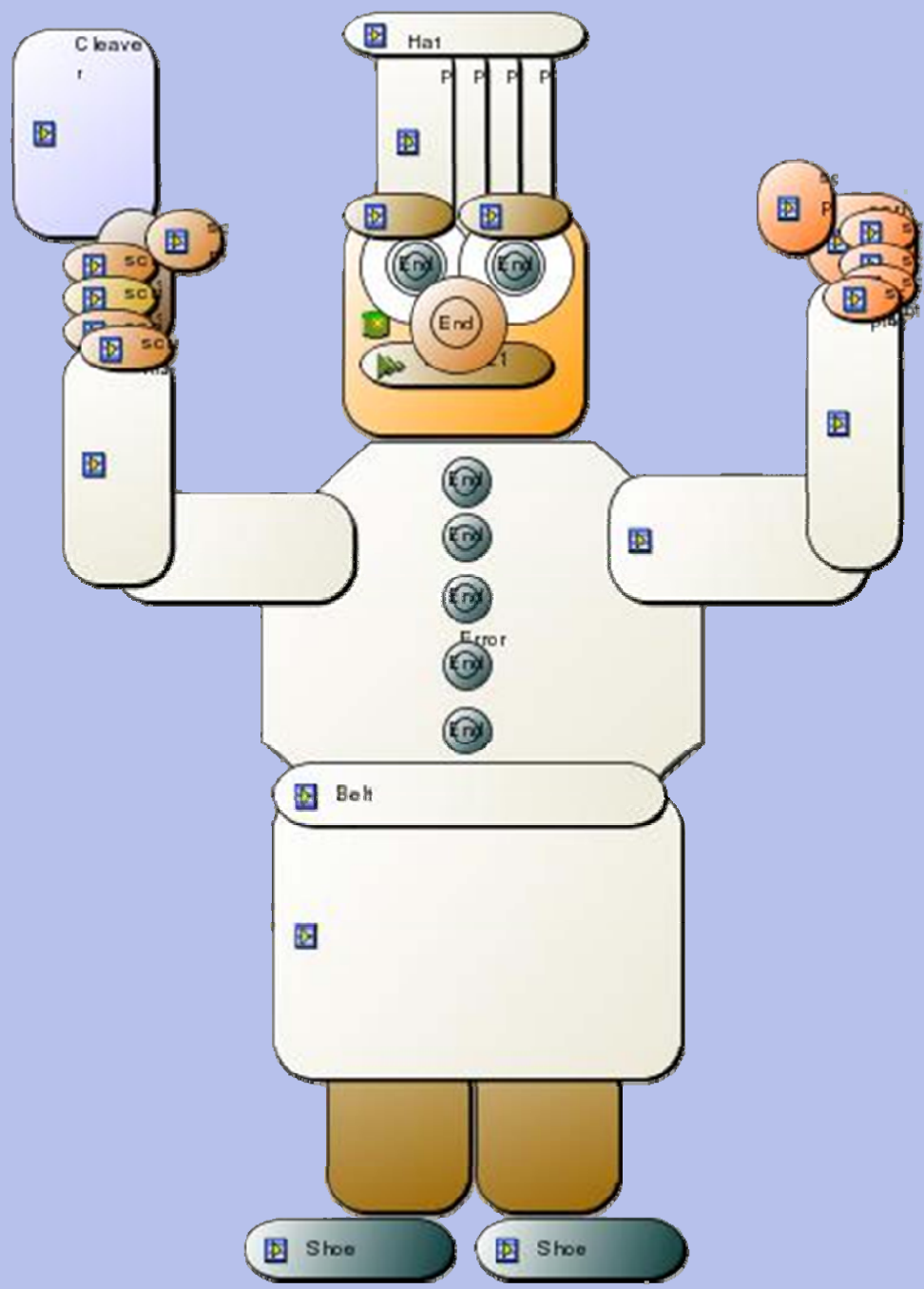
Hidden in
the Introduction

- Asynchrony
 - Sender does not have to wait for receiver to process message
 - Temporal decoupling
- Throttling
 - Receiver can consume messages at its own pace
 - Processing units can be tuned independently
- Can be Reliable Over Unreliable Networks
 - Messages can transparently be re-sent until delivered
 - Think cell phones – intermittent and unreliable
- Insertion of intermediaries (Pipes-and-Filters)
 - Composability
 - Transformation, routing etc.
- Throughput over latency
 - “Wider bridges not faster cars”

A New "Tower of Babel"

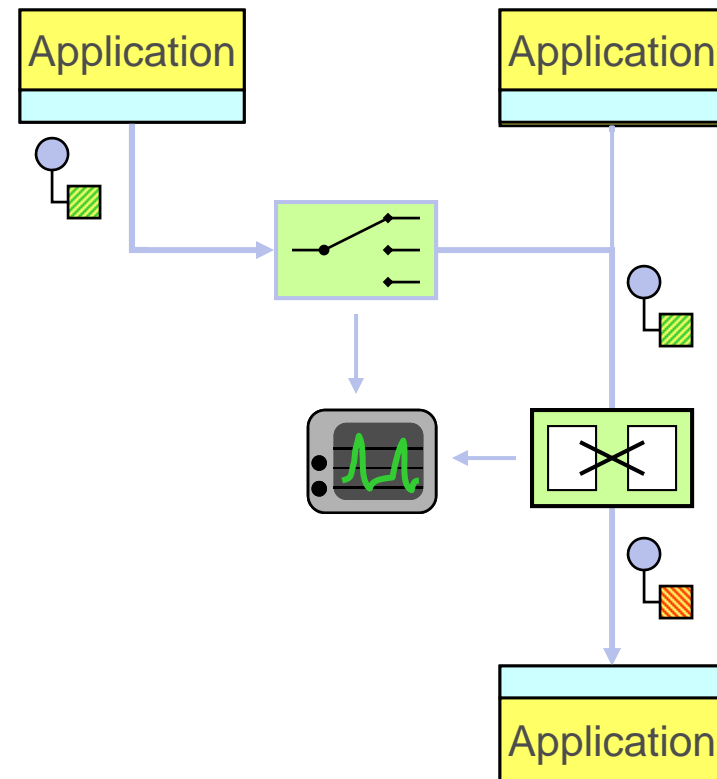


Gartner "Magic Quadrant" for
Integration and Middleware 2001



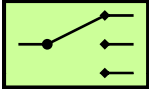
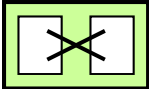
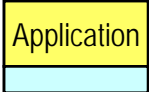
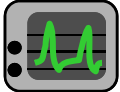


Messaging Pattern Language

1. Transport messages
2. Design messages
3. Route the message to the proper destination
4. Transform the message to the required format
5. Produce and consume messages
6. Manage and Test the System



Messaging Pattern Language

- | | | | |
|---|---|--------------------------------|---|
| 1. Transport messages | ➔ | <i>Channel Patterns</i> |  |
| 2. Design messages | ➔ | <i>Message Patterns</i> |  |
| 3. Route the message to the proper destination | ➔ | <i>Routing Patterns</i> |  |
| 4. Transform the message to the required format | ➔ | <i>Transformation Patterns</i> |  |
| 5. Produce and consume messages | ➔ | <i>Endpoint Patterns</i> |  |
| 6. Manage and Test the System | ➔ | <i>Management Patterns</i> |  |

Welcome to PLoP 2002

- [PloP 2002 Proceedings](#)
- [Call for papers](#)
- [Focus Topics](#)
- [Paper Submissions](#)
- [Schedule](#)
- [Registration](#)
- [Location](#)
- [Call for Volunteers](#)
- [All PLoPs](#)



PLoP 2002 Proceedings (Draft)

Note to authors: Please check the link to the paper and make sure that it contains your final revision. Any corrections should be sent to Weerasak Witthawaskul at plop2002chair@yahoo.com.

Copyright 2002 by paper authors. Permission is granted only to copy for the PLoP 2002 conference.

Update: 9 Sep 2002 Mock Workshop Paper - [Distributed Cache Pattern](#)

Section 1 Accepted Papers

| # | Authors | Title | Shepherd | Program Committee |
|--|---|--|---------------------------------|--------------------------------|
| Plenary Session | | | | |
| 18 | I. Araujo, M. Weiss | Linking Patterns and Non-Functional Requirements (was 'Using the NFR Framework for Representing Patterns') | Brian Marick | Eric Evans |
| Group 2 Leader: Martin Fowler and Ali Arsanjani | | | | |
| 1 | A. Arsanjani | Patterns for Implementing Grammar-Oriented Object Design | Masao Tomono | John Vlissides |
| 3 | A. Arsanjani | Towards a Pattern Language for Web Services Architecture (was 'Patterns for Web Services Architectures') | Gustavo Rossi | John Vlissides |
| 14 | G. Hohpe | Enterprise Integration Patterns | Philip Eskelin | John Vlissides |
| 4 | A. Corsaro, D. C. Schmidt, R. Klefstad, C. O'Ryan | Virtual Component A Design Pattern for Memory-Constrained Embedded Applications | Michael Kircher | Doug Schmidt |

Section 2 Large Pattern Language Group Papers

| Group | Pattern Language | Leaders |
|-------|--|--|
| 1 | Patterns of System Integration with Enterprise Messaging | Bobby Woolf , Kyle Brown |
| 2 | Strategic Design (excerpt from Domain Driven Design) - Entire manuscript can be downloaded from here . | Eric Evans |
| 3 | Some Algorithm Structure and Support Patterns for Parallel Application Programs (abstract) | Berna Massingill , Timothy G. Mattson , Beverly A. Sanders |

<http://hillside.net/plop/plop2002/proceedings.html>

"Enterprise Integration Patterns"
G. Hohpe

"Patterns of System Integration with Enterprise Messaging"
B. Woolf, K. Brown

- OOPSLA 2003
 - 185,000 Words
 - 700 pages
 - 50,000 copies
- Translations
 - English
 - Russian
 - Chinese Traditional
 - Korean
- www.eaipatterns.com
 - Sketches, summaries under Creative Commons
 - Visio, Omnigraffle stencils

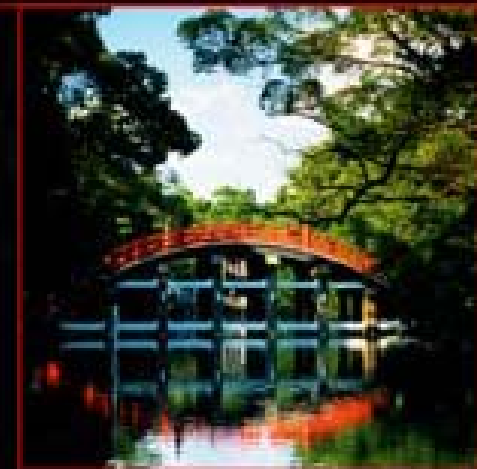
The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

*DESIGNING, BUILDING, AND
DEPLOYING MESSAGING SOLUTIONS*

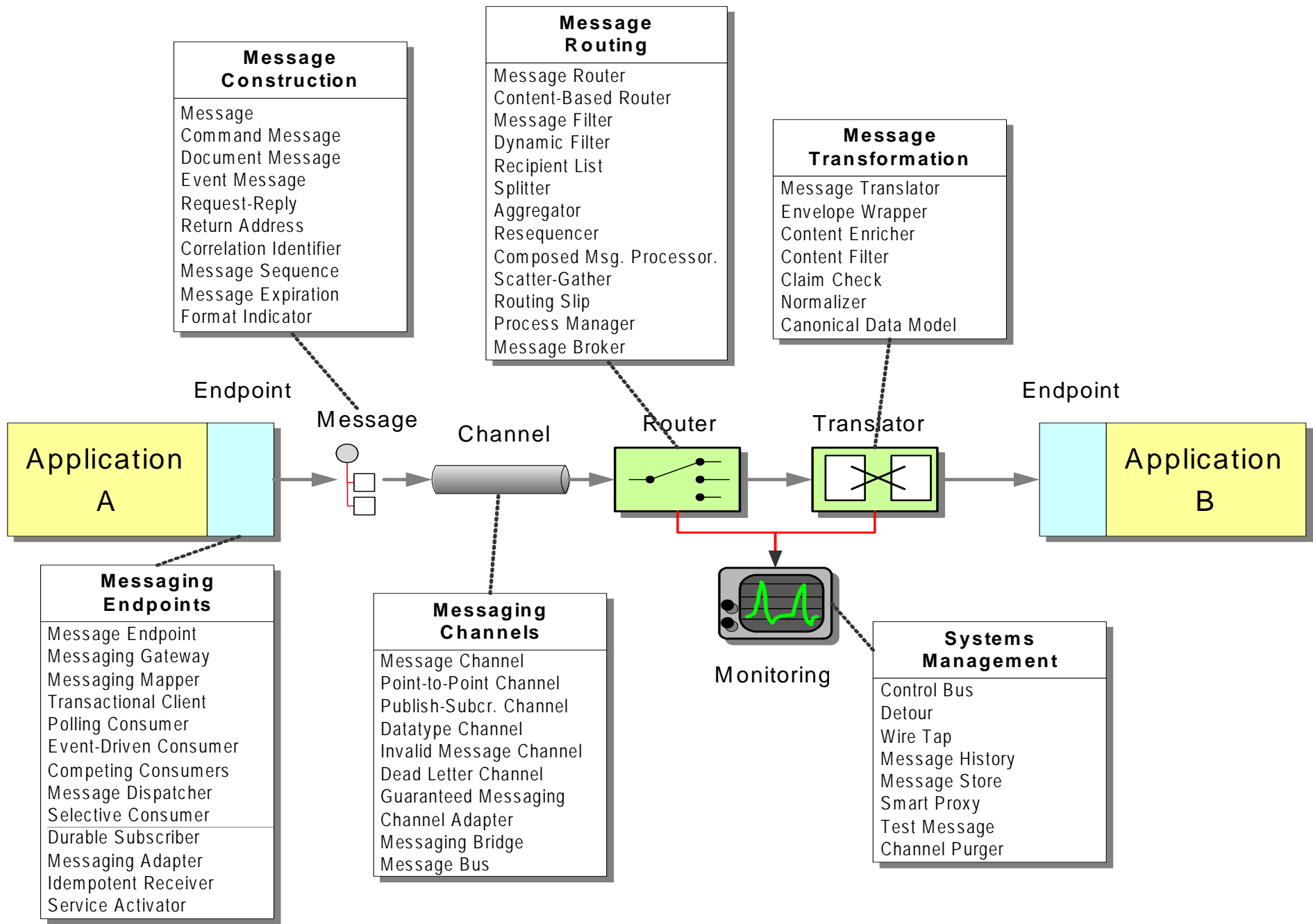
GREGOR HOHPE
BOBBY WOOLF

WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON

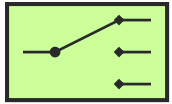


Forewords by John Crupi and Martin Fowler

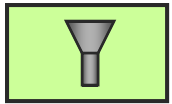




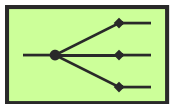
Visual Language



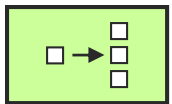
Content-Based Router



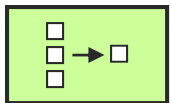
Message Filter



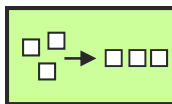
Recipient List



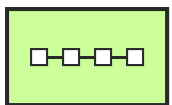
Splitter



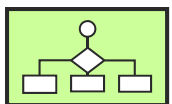
Aggregator



Resequencer



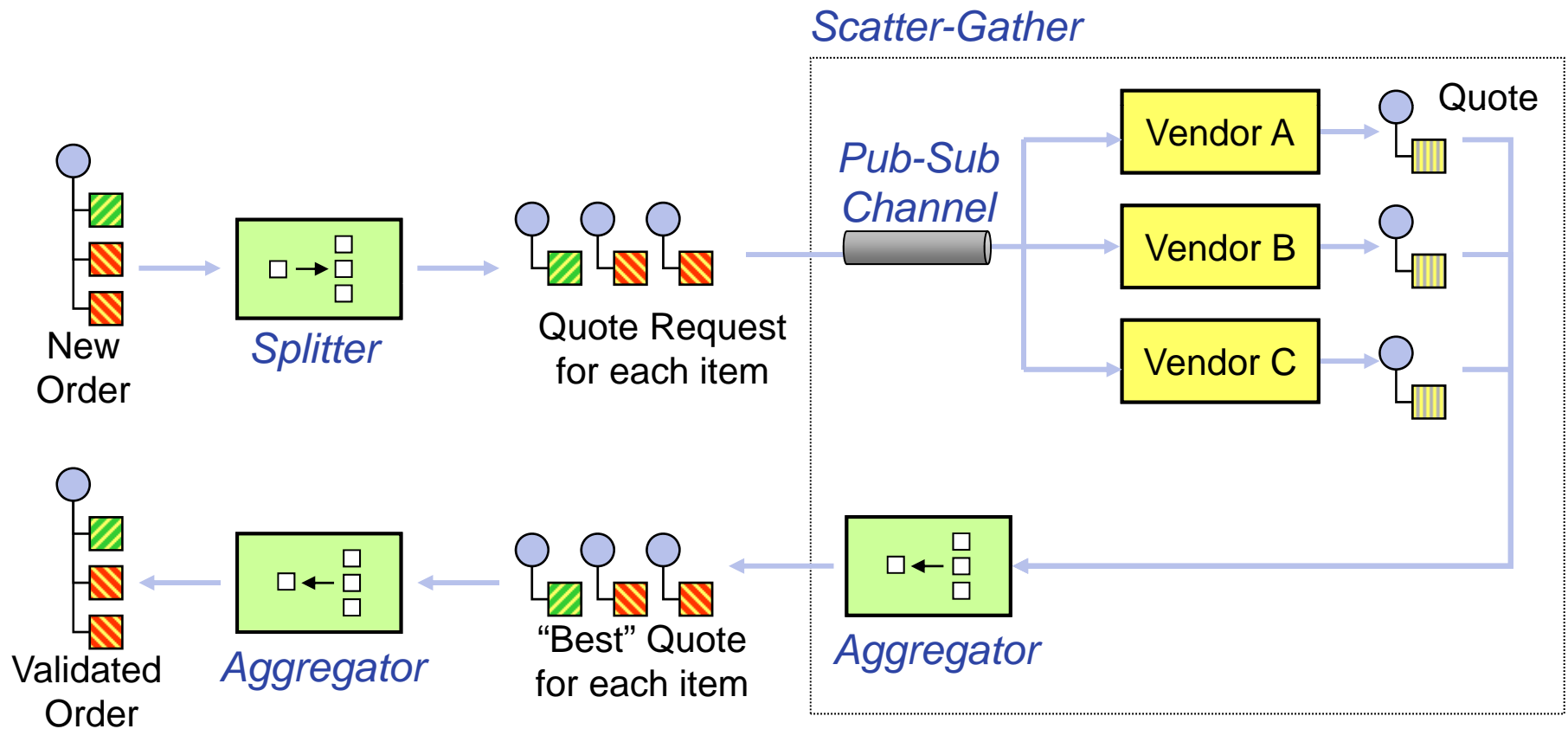
Routing Slip (Itinerary)



Process Manager

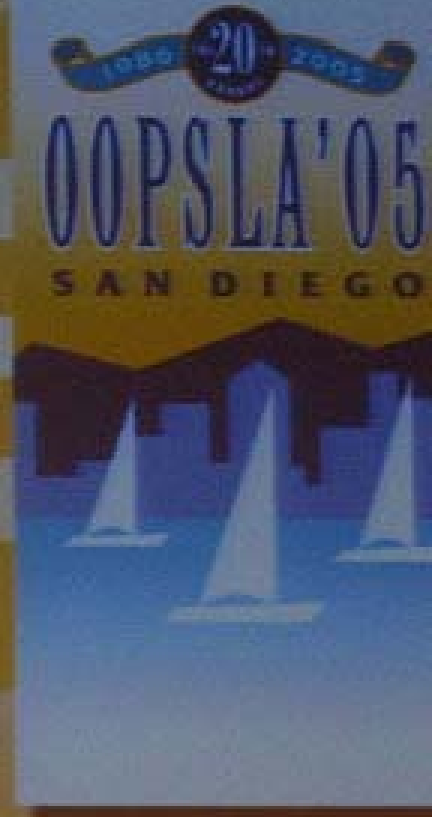
Composing Patterns

- Receive an order
- Get best offer for each item from vendors
- Combine into validated order.



Software Patterns

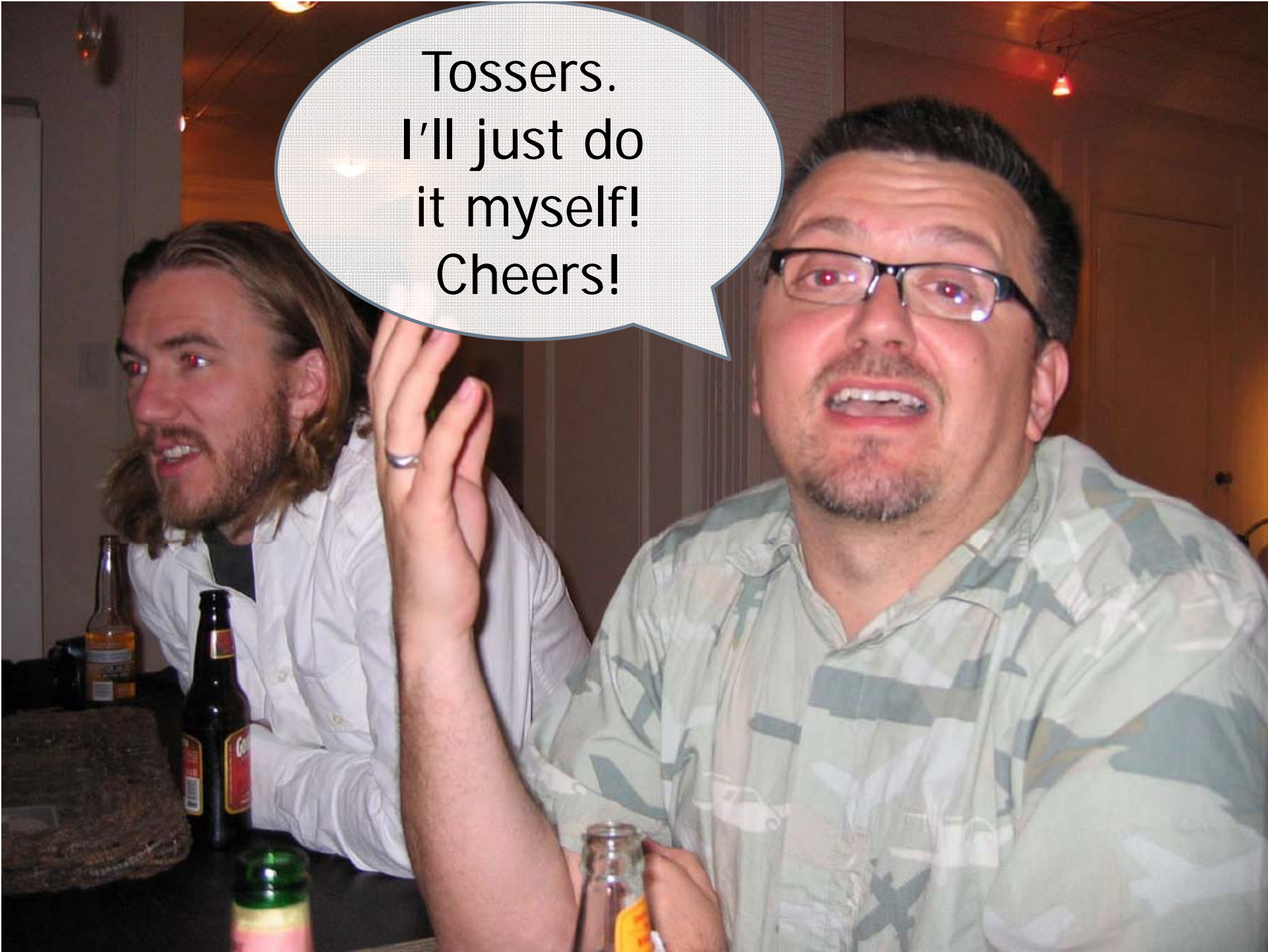
- Buschman, *Pattern-Oriented Software Architecture*
- Dyson, *Architecting Enterprise Solutions*
- Fowler, *Patterns of Enterprise Application Architecture*
- Gamma et al, *Design Patterns*
- Hohpe et al, *Enterprise Integration Patterns*
- Kircher, *Pattern-Oriented Software Architecture*
- Schmidt, *Pattern-Oriented Software Architecture*
- Law/Garland *Software Architecture*
- Strawbridge et al, *Integration Patterns*





Someone
better build
a tool that
supports
these!

Home-made wisdom elixir

A photograph of two men sitting at a bar. The man on the left has long hair and a beard, wearing a white shirt. The man on the right has glasses and a goatee, wearing a camouflage shirt. A speech bubble is overlaid on the image, containing the text: "Tossers. I'll just do it myself! Cheers!".

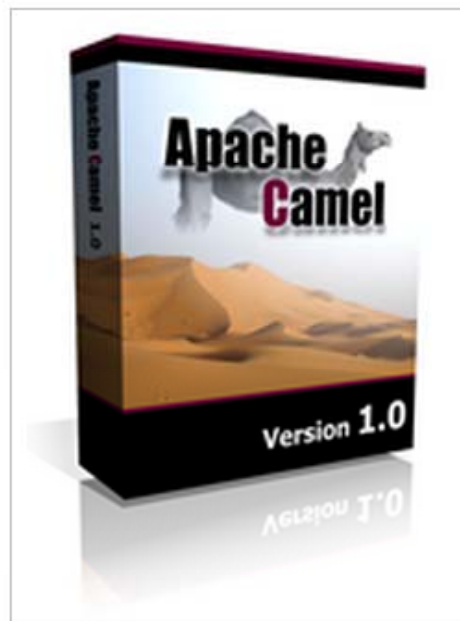
Tossers.
I'll just do
it myself!
Cheers!

James Strachan's Blog

Random ramblings on Open Source, integration and other malarkey

TUESDAY, 15 MAY 2007

➔ Enterprise Integration Patterns in Java using a DSL via Apache Camel



For those of you who missed me rambling about this at JavaOne I thought I'd introduce Camel to you.

[Apache Camel](#) is a powerful rule based routing and mediation engine which provides a POJO based implementation of the [Enterprise Integration Patterns](#) using an extremely powerful fluent API (or declarative [Java Domain Specific Language](#)) to configure routing and mediation rules.

The [Domain Specific Language](#) means that Apache Camel can support type-safe smart completion of routing and mediation rules in your IDE using regular Java code without huge amounts of XML configuration files; though [Xml Configuration](#) inside of [Spring 2](#) is also supported.

A good way to get started is to take a look at the [Enterprise Integration Patterns](#) catalog and see what the Java code of an example looks like. For example, try the [message filter](#), [content based router](#) or [splitter](#).

About Me



James Strachan
Mells, Frome,
England, United
Kingdom

Software Fellow at [FuseSource](#)

[View my complete profile](#)

Links

[FuseSource](#)

[Delicious](#)

[My old blog](#)

Open Source Projects I work on

[Apache ActiveMQ](#)

[Apache Camel](#)

[Apache Karaf](#)

[Apache ServiceMix](#)

[Fuse Fabric](#)

[Scalate](#)



James Strachan
5096 commits, 42 kudos



Camel in Action

Ralf ww_trip10's Profile >

READ ENTRY

Camel IN ACTION

Claus Ibsen
Jonathan Anstey

Forewords by
Gunter Reiser and Alison Spalden

HANNING



Like

Be the

Like

ADD COMMENT

Send a postcard

Patterns Origin – Christopher Alexander

BED ALCOVE

Design problem

Bedrooms make no sense.

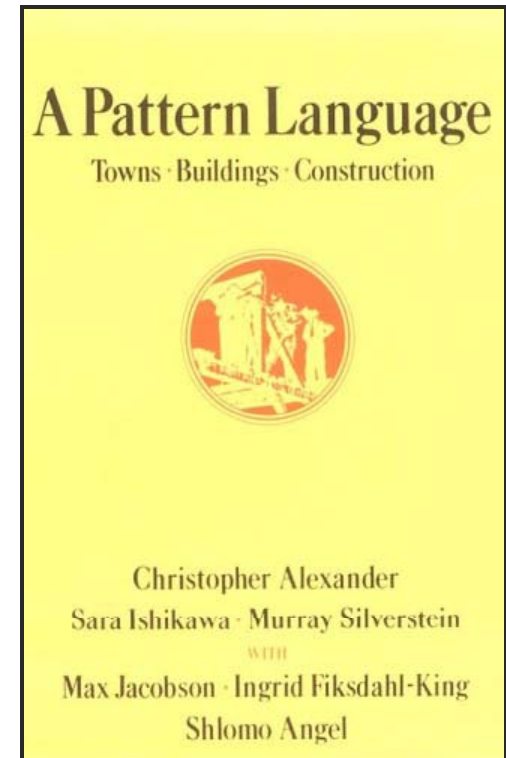
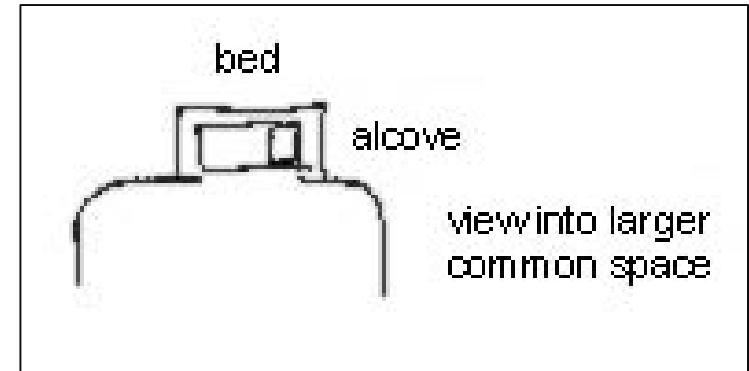
Forces

First, the bed in a bedroom creates awkward spaces around it: dressing, working, watching television, sitting, are all rather foreign to the side spaces left over around a bed. (...)

Second, the bed itself seems more comfortable in a space that is adjusted to it.

Solution

Don't put single beds in empty rooms called bedrooms, but instead put individual bed alcoves off rooms with other nonsleeping functions, so the bed itself becomes a tiny private haven.

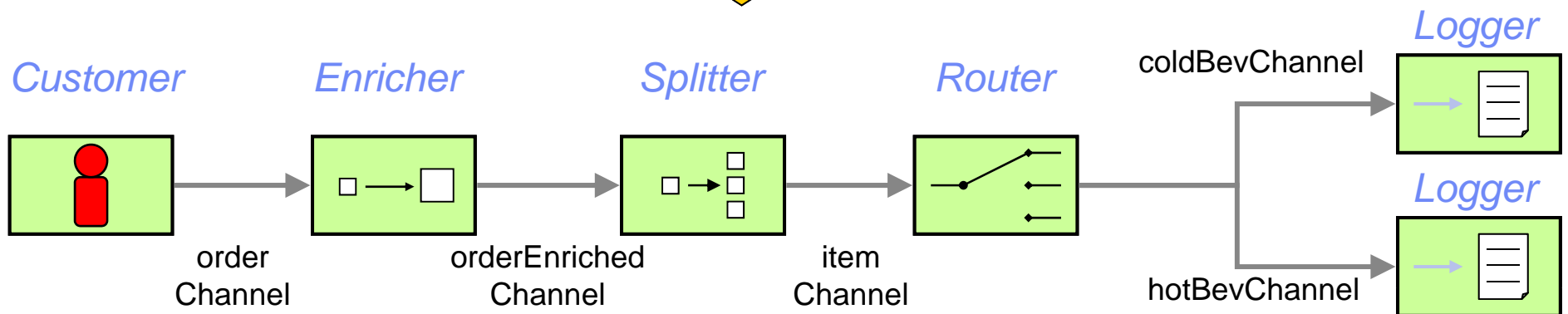
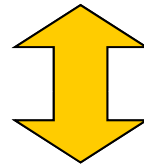


Patterns – 15 Years After GoF

- “Mind sized” chunks of information (Ward Cunningham)
- Human-to-human interaction
- Expresses intent (“why” vs. “how”)
- Observed from actual experience
- New programming models bring new patterns
- NOT:
 - A firm rule –always a time when not to use
 - Copy-paste code
 - Isolated. Part of a Pattern Language

Patterns as Executable Domain Language?

```
CUSTOMER orderChannel
ENRICHER orderChannel orderEnrichedChannel
SPLITTER orderEnrichedChannel itemChannel "/Order/Item"
ROUTER itemChannel coldBevChannel "Item = 'FRAPPUCINO'"
        hotBevChannel
LOGGER coldBevChannel
LOGGER hotBevChannel
```



Patterns

Components



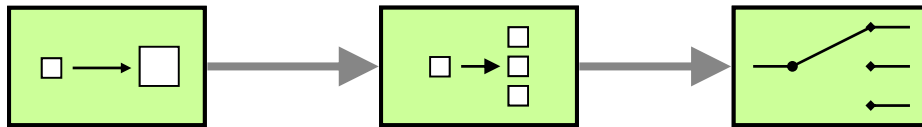
Patterns

- Human communication
- Fuzzy
- Design tool
- Platform independent

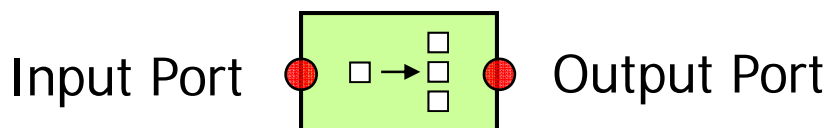
Components

- System Communication
- Precise
- Executable
- Platform dependent

-
- Pipes and Filters style: simple composability

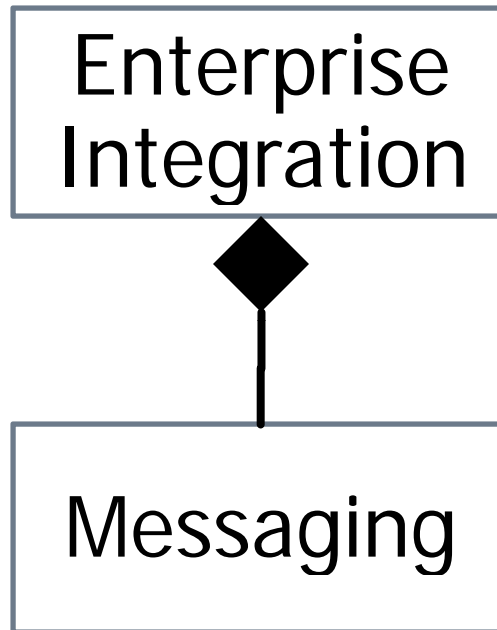


- Easily formalizable: input ports, output ports

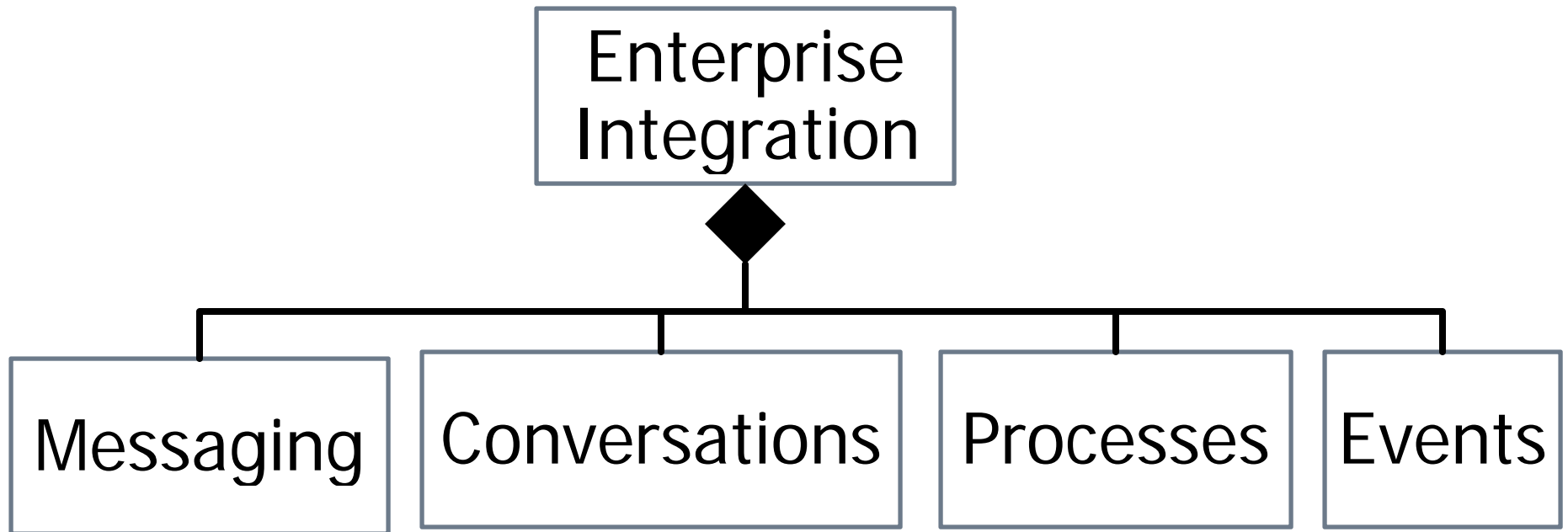


- Leverage other domain languages, such as XSLT

Enterprise Integration or Messaging?

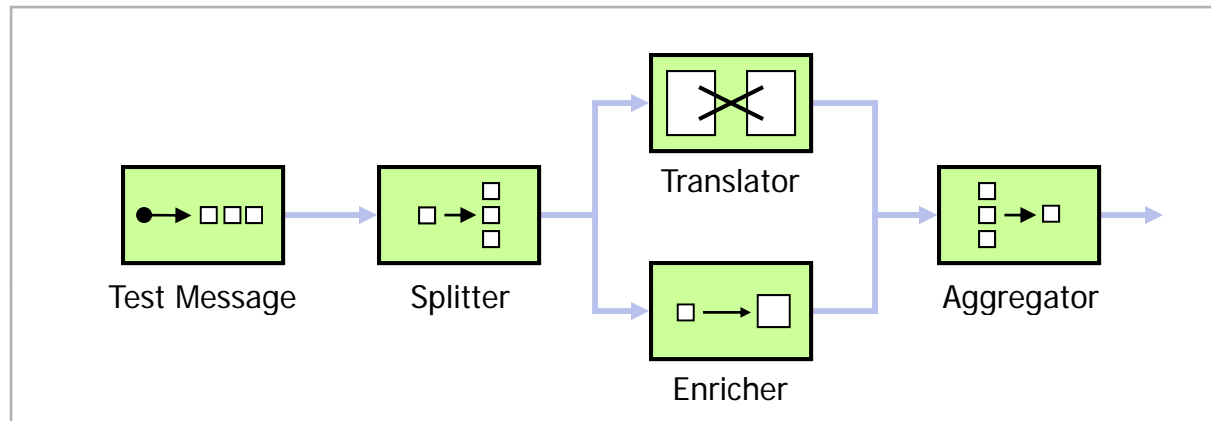


Enterprise Integration or Messaging?



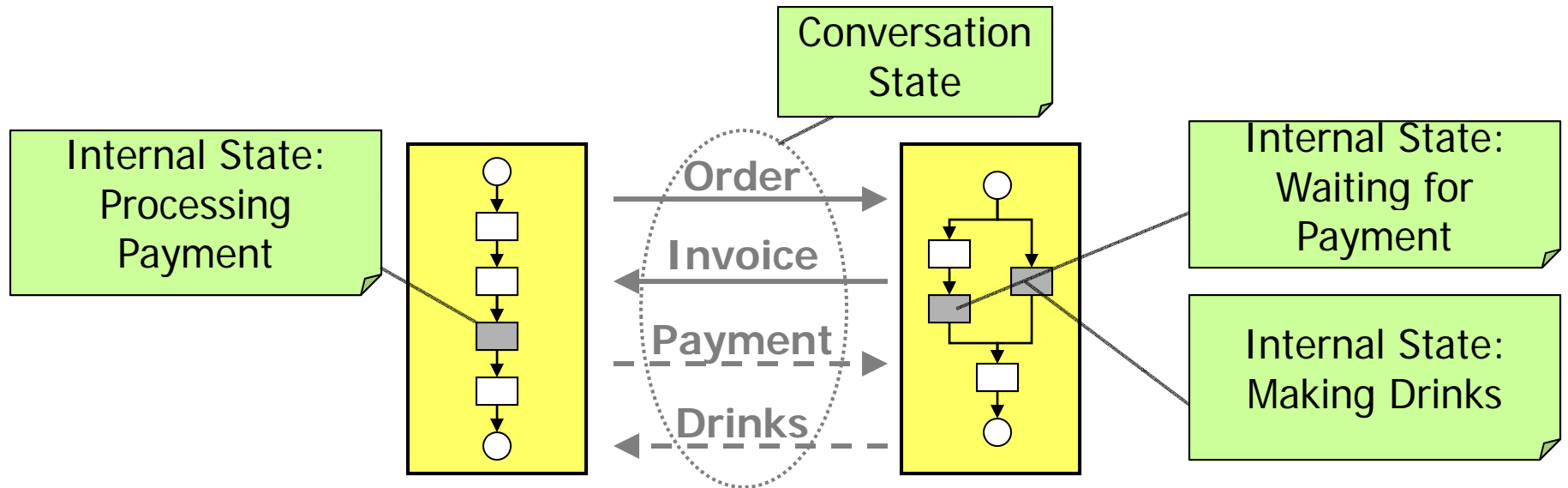
Messaging

Flow of messages through processing nodes



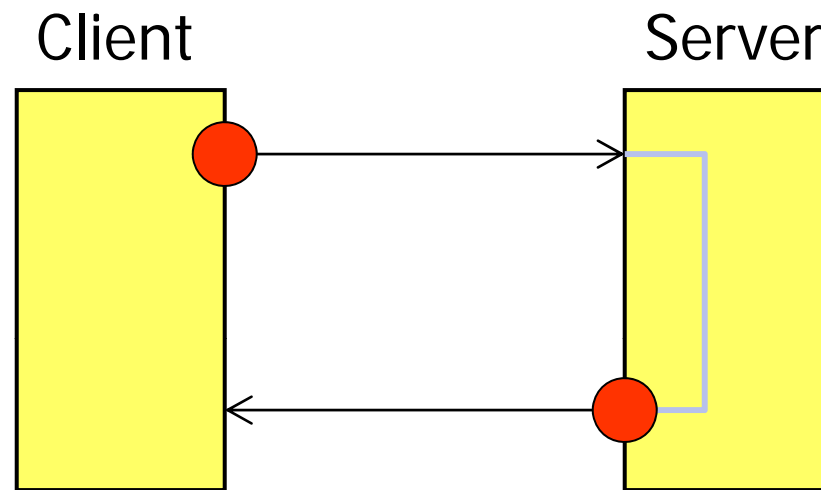
- Stateless -> scaleable, decoupled
- Error handling?
- Complex interactions (no guarantees)

Conversations



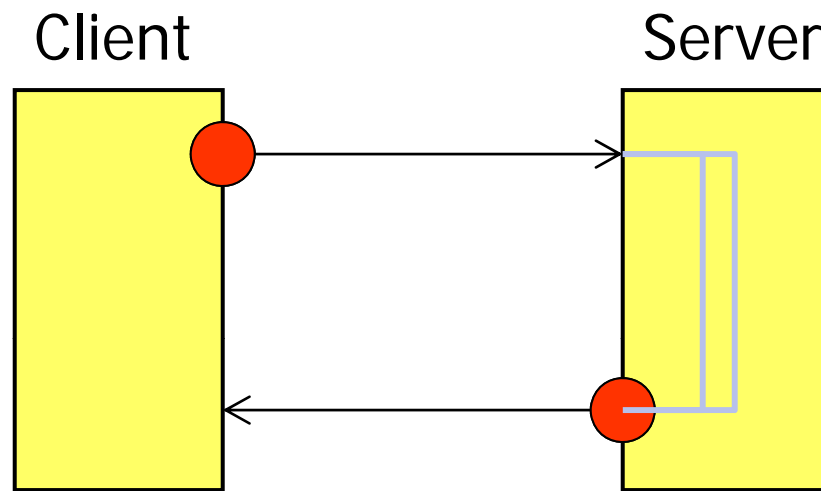
- Each conversation corresponds to one process instance
- Each participant has a (potentially different) process definition

Simple Example: Request Response

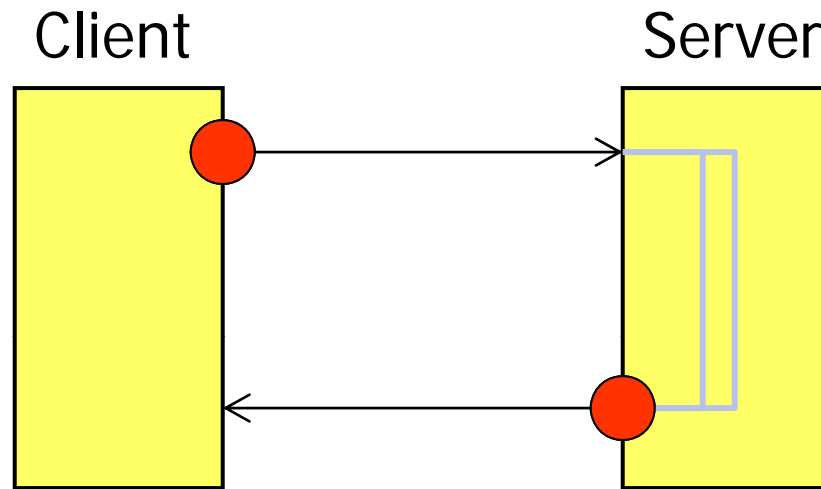


Conversation = Series of Related Messages

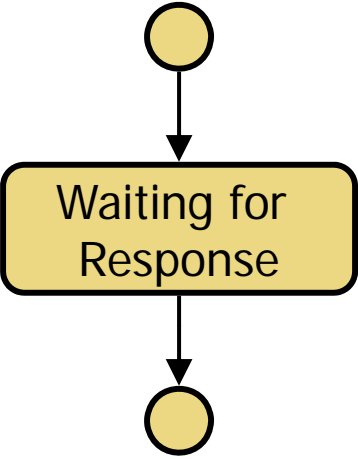
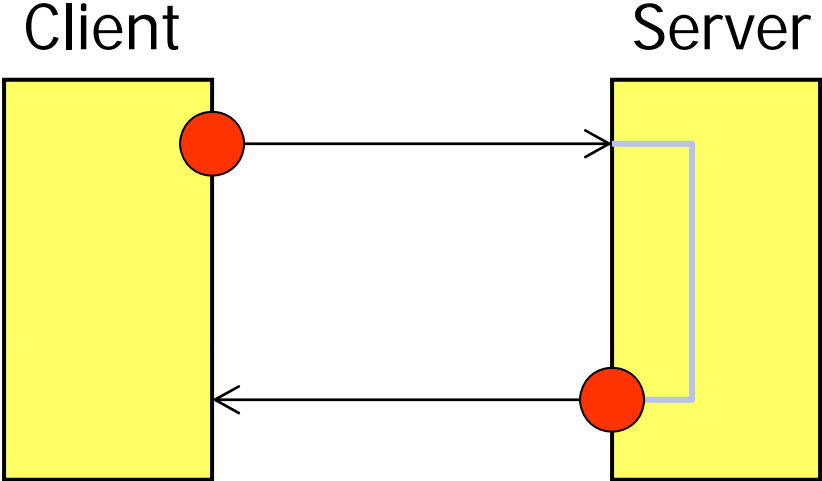
Response Message Lost



Response Message Delayed



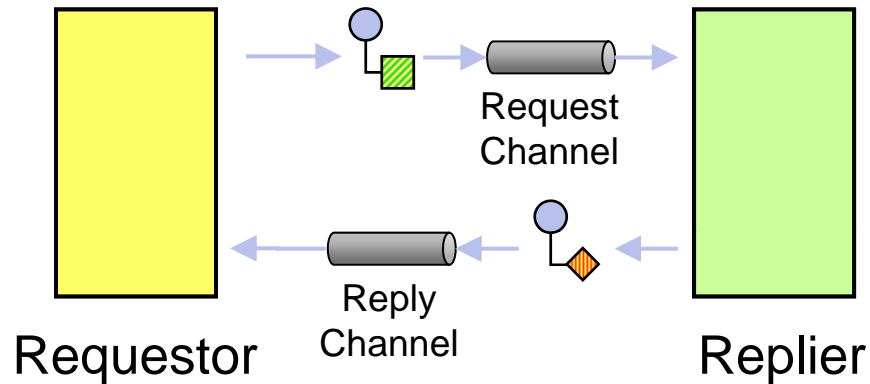
Conversation State



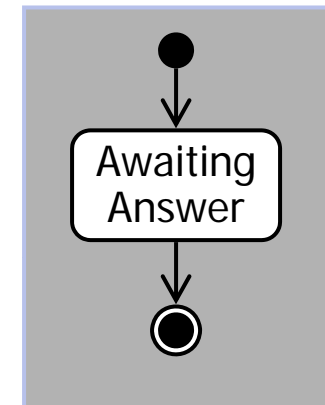
Challenges: Describing Conversations

- Sequence Diagrams (UML 1.x) only show one instance, not rules of interaction
- Sequence Diagrams (UML 2.0) more powerful, but non-intuitive notation
- WS-CDL describes conversation state, but very little adoption
- WS-BPEL a little verbose, looking from participant perspective
- Temporal Logic expressive, but not good for sketch

Request-Reply

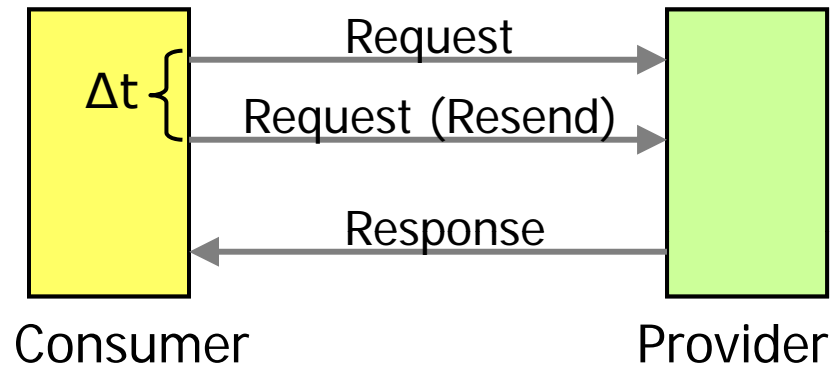


Conversation
State Chart



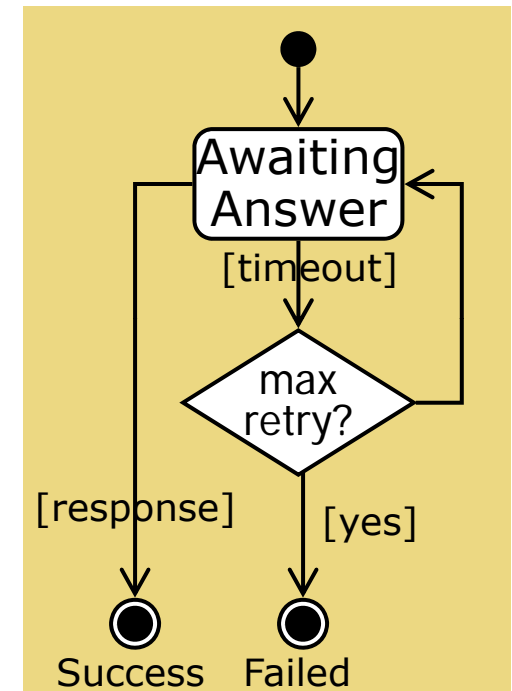
- Simplest conversation
- Single Conversation state: waiting for reply, complete
- Gets more complicated once error conditions considered

Request-Reply with Retry



- Sender can repeat request n times
- Provider has to be idempotent
- Receiver also has to be idempotent
- Example: RosettaNet Implementation Framework (RNIF)

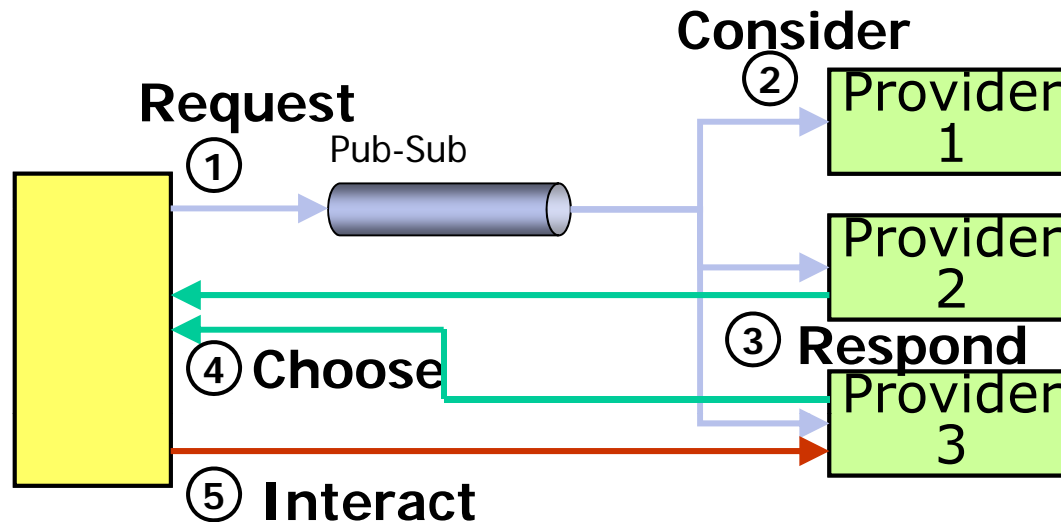
Conversation State



How can a service find a conversation partner in when it has no knowledge about the network and its services?

-
- Point-to-point communication requires knowledge of the conversation partner (or channel).
 - The late binding between a service consumer and the service endpoint lowers the location coupling between them.
 - Discovery may be on the critical path to establishing a conversation.
 - Even in the presence of a central lookup service, a new participant has to first establish a connection to the lookup service.

Dynamic Discovery

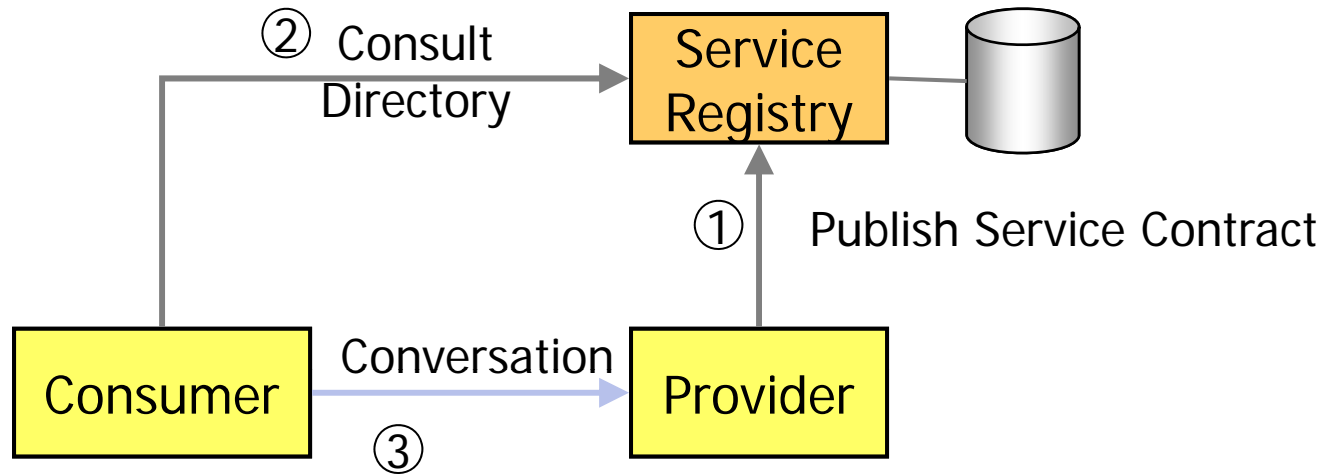


1. Broadcast request
2. Provider(s) consider whether to respond (load, suitability)
3. Interested providers send responses
4. Requestor chooses "best" provider from responses
5. Requestor initiates interaction with chosen provider
 - Examples: DHCP, TIBCO Repository discovery

How can a service find a conversation partner across a large network without flooding the network with requests?

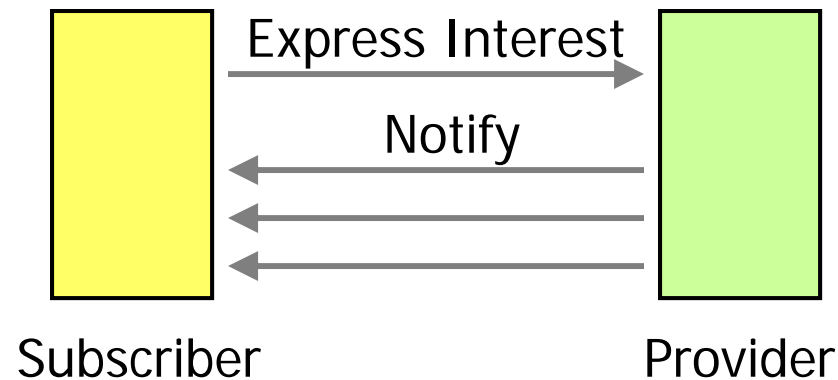
-
- The late binding between a service consumer and the service endpoint lowers the location coupling between them.
 - Discovery may be on the critical path to establishing a conversation.
 - Many networks do not route broadcast packets beyond the local network.
 - Often centralized administration is involved in setting up a new service.

Consult Directory



- Directory may store additional metadata about the service
- "Match making based on"
 - *Unique Identifiers*
 - *Interface Definition / Type*
 - *Attributes*
 - *Keyword match*

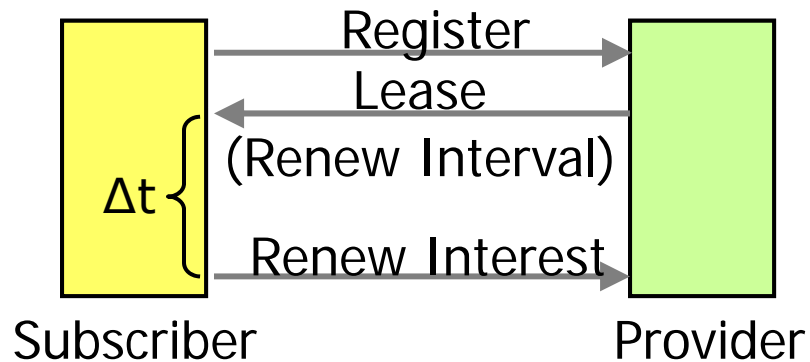
Subscribe-Notify (Multi-responses)



-
- Subscriber expresses interest in receiving notifications
 - Subscriber receives messages until a stop condition is reached:
 - Subscriber sends a stop request
 - A deadline is reached without the subscriber renewing interest
 - Subscriber does not respond to requests from provider
 - Provider notifies subscriber of end of transmission

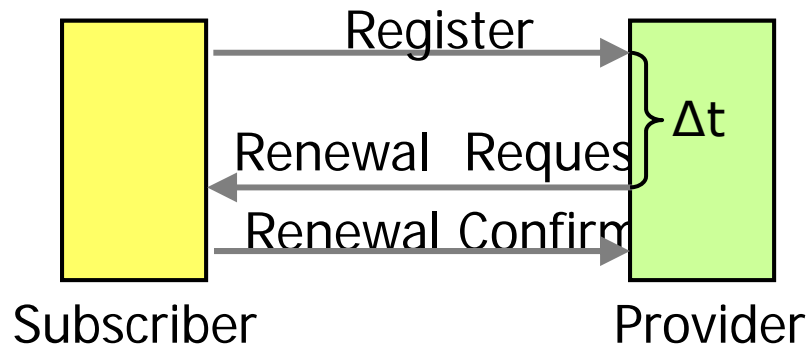
Renewing Interest

Automatic Expiration



- "Lease" model
- Heartbeat / keep-alive
- Subscriber has to renew actively
- Example: Jini

Renewal Request



- "Magazine Model"
- Subscriber can be simple
- Provider has to manage state for each subscriber

Conversation Pattern Language

Messages

- Initiating Message
- Follow-on Message
- Complete Message
- Side Conversation (Sub-Conversation)
- Acknowledgment Message

Simple Conversations

- Reliable Delivery
- Sync Request-reply
- Async. Request-Reply messages
- Async. Request-Poll for result
- Subscribe-Notify
- Tacit Agreement
- Reaching agreement

Conversation Pattern Language

Coordinated Conv.

- Vote / Poll
- Reaching Agreement / Two-phase vote
- Unanimous agreement

Establishing Conv.

- Discovery
- Introduction
- Three-Way Handshake
- Role negotiation
- Establishing trust

Renewing Interest

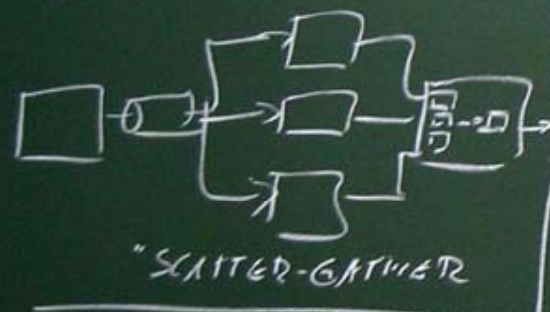
- Lease / Automatic Expiration
- Renewal Reminder

Exception Handling

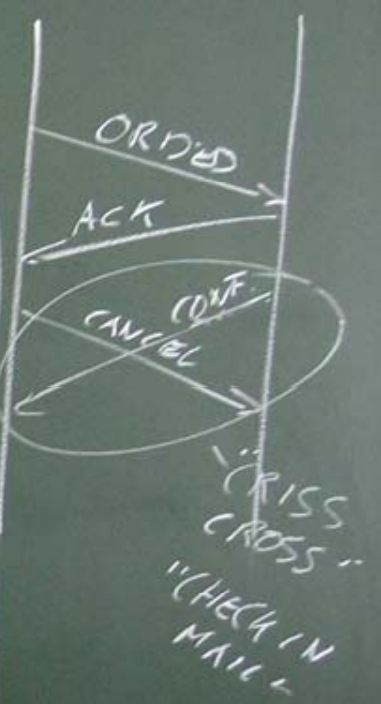
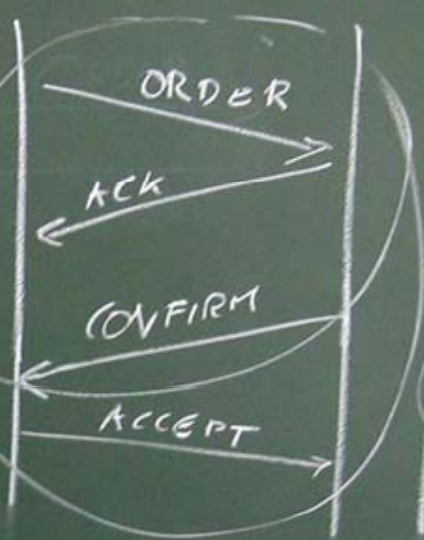
- Two-phase Commit
- Compensation Action
- Retry / Resend (Idempotent Receivers)
- Write-Offs



- Ability to Cancel
- Ack before Proc.
- Non repudiation (Ack), despite reliable msg

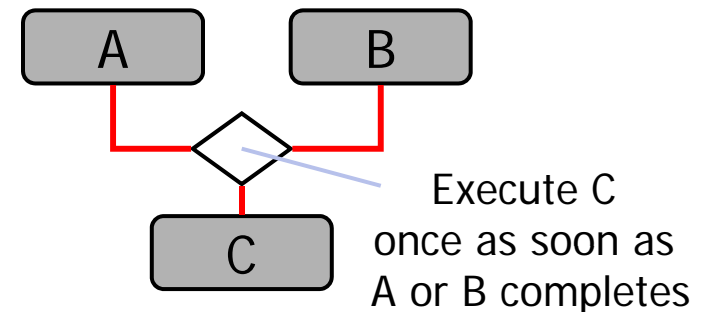
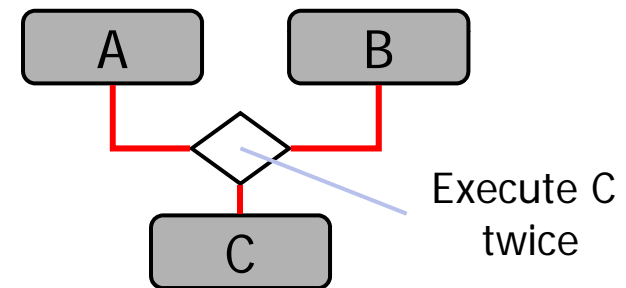
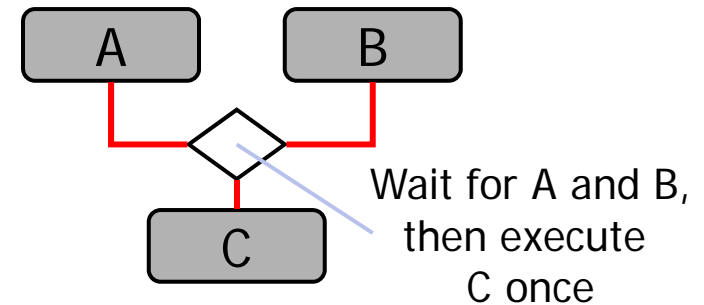


"Counter-offer"



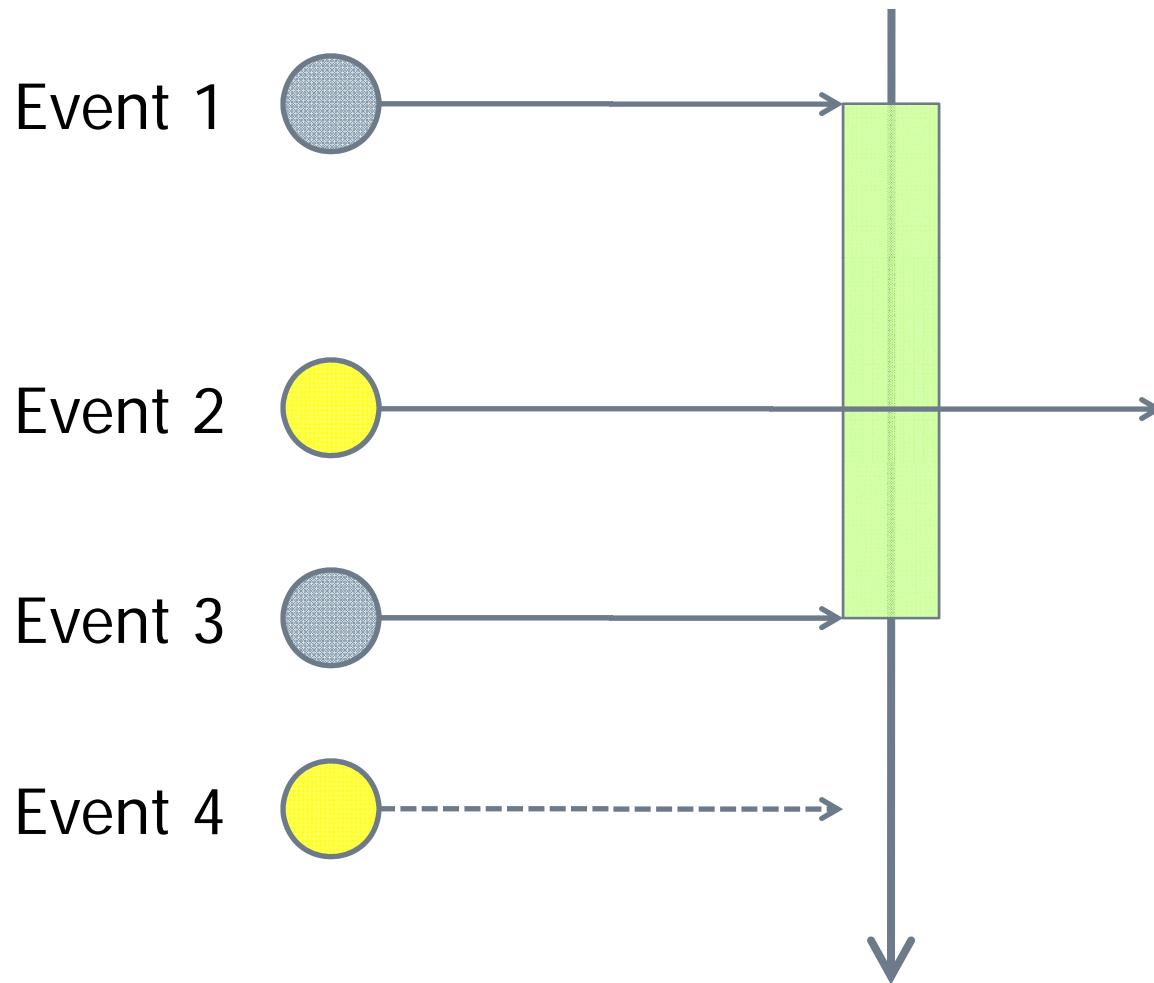
Workflow Patterns - Example

- Synchronizing Merge
 - Merge many execution paths. Synchronize if multiple paths are taken.
- Multiple Merge
 - Merge many execution paths without synchronizing.
- Discriminator
 - Merge many execution paths without synchronizing. Execute the subsequent activity only once.



Event Pattern - Example

“Window of Opportunity”

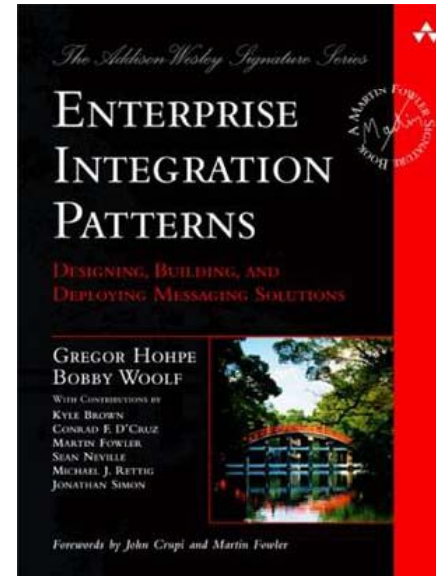


- Messaging Patterns (65)

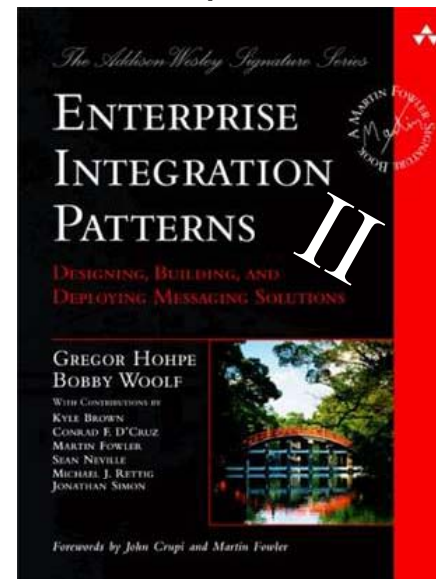
- Messaging Systems
- Messaging Channels
- Message Construction
- Message Routing
- Message Transformation
- Messaging Endpoints
- System Management

- Conversation Patterns

- Discovery
- Establishing a Conversation
- Multi-party Conversations
- Reaching agreement
- Resource Management 50
- Error Handling



www.eaipatterns.com



www.conversationpatterns.com

Photo Credits

- <http://nevermoregraphix.deviantart.com>
- http://www.travelpod.com/members/ralf_ww_trip10
- <http://www.flickr.com/photos/mukluk>